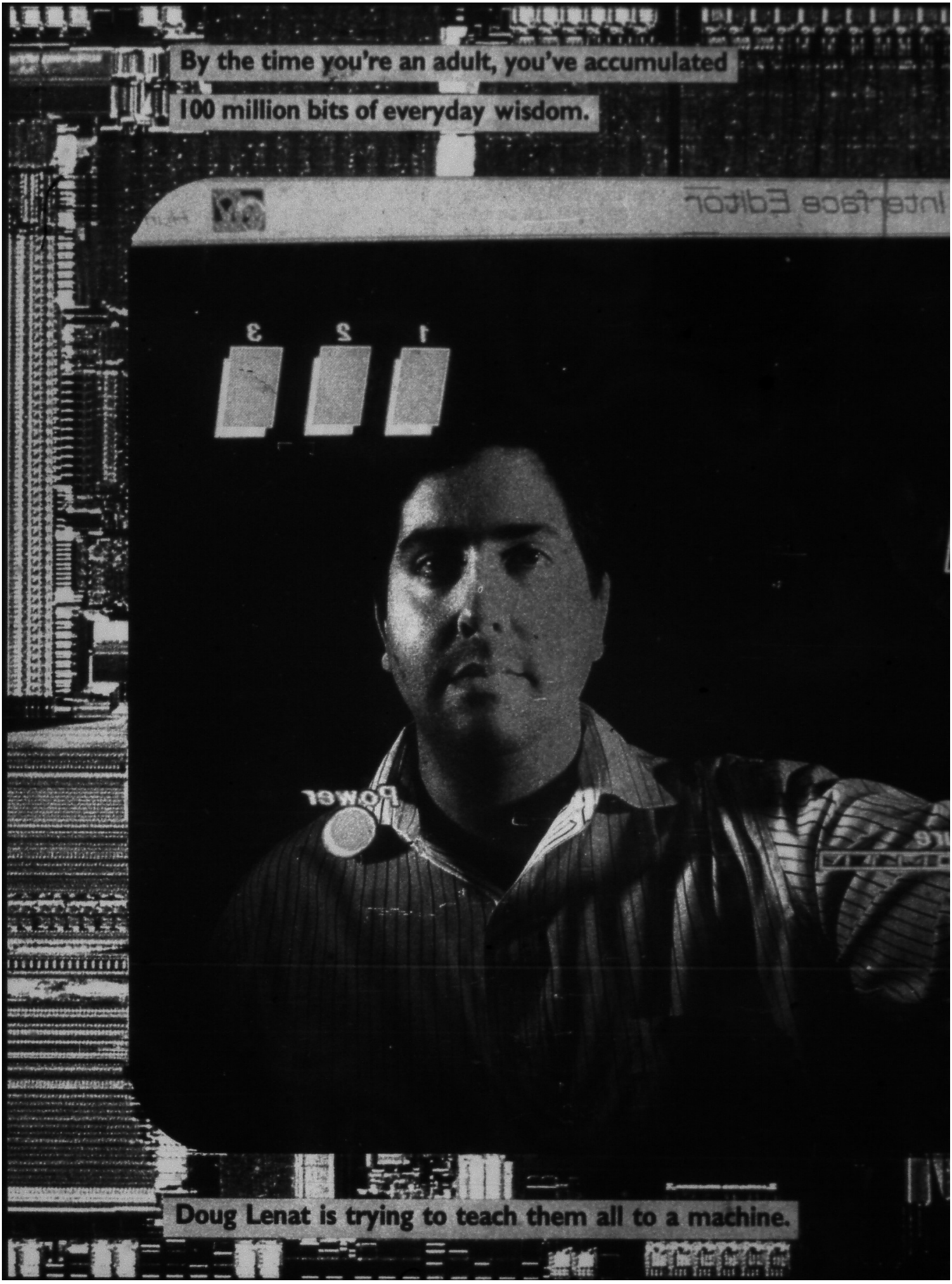
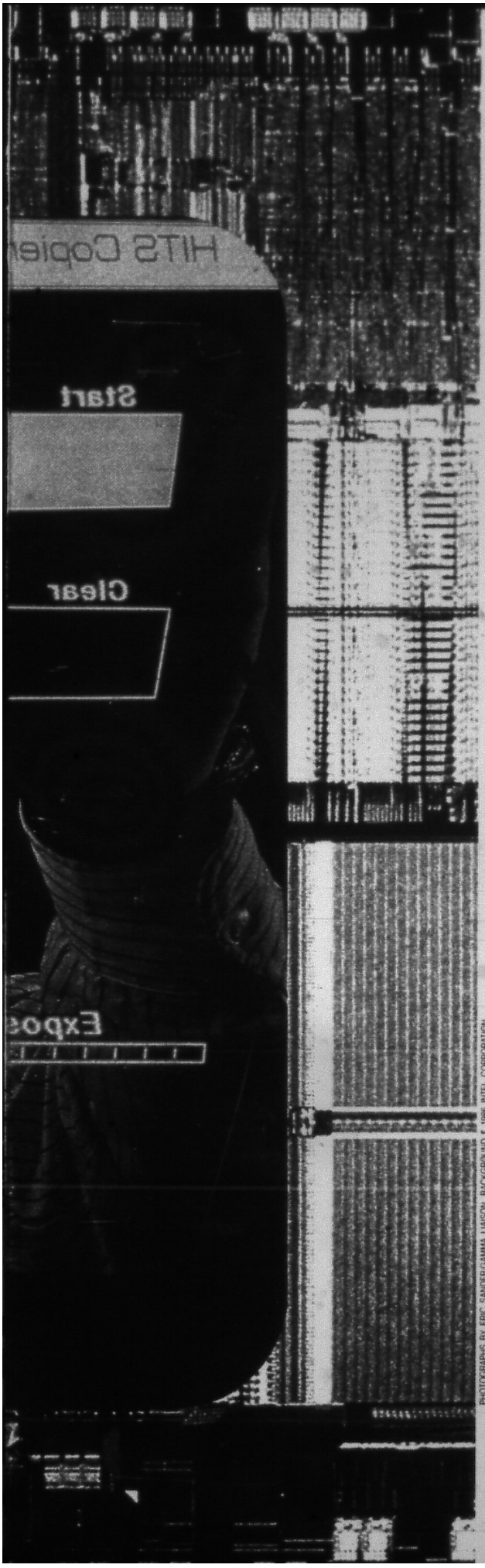

By the time you're an adult, you've accumulated
100 million bits of everyday wisdom.
Doug Lenat is trying to teach them all to a machine.



PHOTOGRAPHS BY ERIC SANDER/GAMMA LIAISON. BACKGROUND © 1986 INTEL CORPORATION

COMMON SENSE AND THE COMPUTER

BY DAVID H. FREEDMAN

Karen Pittman is hunched over a copy of *How to Troubleshoot and Repair Any Small Gas Engine*. It is not a gripping read. Pittman doesn't care much about the subject, and when she finishes she will probably be no more able to repair a balky lawn mower or outboard motor than she was when she began. What she's looking for in the arcane volume is something subtler: the bits of everyday knowledge that the authors didn't bother to explain to the reader at all—what a lawn is, for instance, or where you're likely to find one.

Every complex repair rule in the book may be predicated on dozens of such commonsense scraps, tidbits of general knowledge that are in need of no explanation. After Pittman collects them all, she must explain every one to a computer. At the end of the session, the computer still won't know a fraction of what a mechanic knows about small engines. But it will know everything Pittman knows about them.

The computer's data base is filled with such mundane stuff: what to do with a knife and fork, what cats and dogs are, where the state of Texas is. There's a little art and biology and history thrown in, but nothing that the average joe wouldn't know.

This educational effort is the brainchild of Doug Lenat, a computer scientist whose friendly manner and roly-poly appearance belie a remarkable intensity. That energy has been crucial in driving a project so ambitious that most of Lenat's fellow researchers have maintained that it couldn't possibly meet its goal: to equip a computer with all the general knowledge of the average adult. Considering that until now no computer has ever absorbed the knowledge of even the average four-year-old, this is a daunting task. But Lenat insists that, as he rounds out the sixth year of his ten-year project, he's right on schedule.

"Teaching a computer all the ordinary things human beings take for granted is not an easy job," he says. "We have to put in ten times as much information as I thought we would; but we're doing it ten times as fast."

Computers, for the most part, are fantastically stupid things. Oh, they seem smart enough: no human brain could ever hope to compute pi to a billion decimal places, or contemplate several million chess moves and their possible consequences. But take a computer beyond these narrow areas of genius—ask it, "What is a tree?" or "Who is taller, a mother or her infant son?" or "If you paint a chair blue, is it still the same chair?"—and the same machine is completely lost.

In Austin, Doug Lenat is giving a computer the gift of common sense with a revolutionary program dubbed Cyc, as in *en-cyc-lopedia*.

Computers are utterly ignorant of the trivial knowledge humans spend a lifetime accumulating. If electronic brains are ever to get beyond the glorified bookkeeping and computing tasks to which they're largely confined today, they need programs that include at least a portion of this unglamorous data base.

One approach toward this end has involved the development of "expert systems." Unlike standard programs, which tell a computer precisely what to do, an expert system incorporates heuristic knowledge—rules of thumb that come only through experience. For example, an auto-repair expert system might be loaded with a few hundred if-then rules such as, "If the engine won't turn over and the temperature is below 20 degrees, then check the battery strength." After checking to see which of the conditions have been met, the computer employs the correct solution. A top-notch expert system can consider thousands of rules and come up with the same results that a human expert would. But expert systems, according to Lenat, are "brittle."

"They have only a veneer of intelligence," he says. "When they're faced with a novel or unexpected situation, they'll give the wrong answer, and be convincing about it." To be wrong less often, computers need to be equipped with "consensus knowledge"—knowledge most of us take for granted. So Lenat and his team are entering 100 million morsels of general information into a program named Cyc (as in *en-cyc-lopedia*). And that's just for starters. In a few years, Lenat claims, Cyc will understand English well enough to continue its education on its own by reading books, newspapers, and magazines and discussing them with human tutors. Ultimately, he asserts, Cyc could serve as a kind of knowledge fountainhead for other computer programs, allowing them to tap into a generalized data base and apply what they learn to their own particular areas of expertise.

"My hope," says Lenat, "is that by 1999 no one would think of buying a computer without Cyc, any more than anyone would now think about buying a computer without word processing."

Lenat could easily pass for the quintessential computer nerd. "Uh-oh," he says, when a video-game-like shriek erupts from one of the computer terminals next to his desk. With the quickness of a parent answering a baby's cry, he pushes off on his black Reeboks, spins his chair around, and attacks the keyboard with the blurred-key-click speed that programmers acquire from being impatient to get instructions into the computer. "Okay, we're in business," he says to the machine.

For someone who is on such intimate terms with computers, Lenat learned his craft in a rather informal way. Most of his college training came from summer programming jobs in computer firms. It wasn't until he was ready to graduate from the University of Pennsylvania in 1971 that he took his first computer-related course. At the time, Lenat was polishing off the requirements for two undergraduate degrees—one in mathematics and one in physics—along with a master's degree in applied mathematics. Then he stumbled upon a course in artificial intelligence.

For the first year Lenat hardly went near a keyboard. Why bother trying to teach Cyc what a truck is if it doesn't understand cars or wheels?

"I had already applied to graduate schools in math and physics," he recounts. "But AI was so fascinating that I decided that's what I wanted to do."

Lenat made good in his new field at Stanford's celebrated computer science department. One of his earliest successes was a program called AM, a "machine learning" program designed to derive new ideas from a preprogrammed data base. Machine-learning programs are essentially computer contemplators: given a set of principles about, say, math or chemistry or baseball, the program turns the data this way and that to see whether it can come up with any new rules.

We use this kind of reasoning every day: the chemist who invents a new polymer or the mechanic who designs a new engine is not so much creating new knowledge as manipulating what's already there. Lenat armed AM with such basic mathematical information as how to determine whether two sets are equal and how to perform the inverse of a mathematical function. When he turned the computer loose, it proceeded to make 700 "discoveries."

AM had noticed, for example, that there was a certain subset of numbers that had unusually few divisors. Then it found that when it multiplied these numbers together in different combinations, it could find any whole number. Thus the program stumbled upon prime numbers and the unique factorization theorem.

AM even uncovered a couple of mathematical principles that appeared to be genuinely original, including one that describes the relationship between the size of a prime number and the number of times it appears as a factor of certain larger numbers. (As it turns out, a human mathematician had already made this discovery, but at the time Lenat—and his program—did not know this.) AM earned the 25-year-old Lenat a prestigious AI award and sudden widespread recognition in the field.

After receiving his doctorate and picking up faculty positions at Carnegie-Mellon and Stanford, Lenat went on to develop other machine-learning programs. But even the best of them kept running out of steam after an initial burst of discoveries. "The learning you get out of these programs is really only what you preengineer into them," Lenat says. "It's like a spring unwinding; after a while the initial programming is exhausted and the computer needs to be given more data. In 1983 I got this vision about what was holding back the program: it lacked common sense."

In 1984 Lenat received an offer he couldn't refuse, as he puts it, to attempt to put his new theory into practice. The offer came from the Microelectronics and Computer Technology Corporation (MCC) in Austin, a research and development consortium funded jointly by Digital Equipment Corporation, Kodak, Apple Computer, NCR, and Bellcore. Lenat moved to Texas and immediately set to work to make the most out of the ten years and the $35 million MCC provided him.

Despite MCC's conservative corporate

PHOTOGRAPH - 1990 INTEL CORPORATION

YES, BUT DO PUTTYTATS FLY?

A typical computer program is a list of commands that tells a computer what to do. A simple addition program might tell the computer something like this:

```
ASSIGN TWO NUMBERS
THE NAMES X AND Y
LET C = X + Y
PRINT C
```

This works fine for such rudimentary tasks as accounting. But an expert system—a program that is equipped with an expert's knowledge of one field, like aeronautics or auto repair—needs a bit more.

The basic expert program consists of two parts. The first, called a knowledge base, is a collection of facts and rules. The second, called an inference engine, is a series of miniprograms that tell the system how to manipulate the facts and rules in a useful way. Without the inference engine, the knowledge base would just be a meaningless jumble of words.

Cyc also consists of a knowledge base and an inference engine. Here's how the two parts might work together to answer the question, "Can Tweety fly?"

Suspecting that knowledge about Tweety might come in handy someday, a Cyc knowledge enterer sits down at a terminal and types in:

```
(INSTANCE-OF TWEETY BIRD)
```

The inference engine understands this to mean that a thing called Tweety is an example of a thing called Bird. Searching through the knowledge base, the inference engine finds that Tweety is not among the things known. It sets aside a small portion of the knowledge base to store the information (Bird Tweety), leaving room to add knowledge about Tweety that it may acquire later. The inference engine labels this new frame of information Tweety and adds Tweety to the knowledge base's list of things it knows about. No further action is taken.

Later someone decides to find out if Cyc is able to draw any conclusions about Tweety. That person types:

```
(ASK (CAN-PERFORM TWEETY FLYING))
```

This code instructs the inference engine to determine whether something called Tweety can fly. Cyc knows what a Tweety is, but can it figure out what a Tweety does?

The inference engine first searches through the list of things the data base knows to see whether Tweety is there. It finds Tweety listed, and so it looks at the Tweety frame. It finds that Tweety is a bird but sees nothing about the ability to fly. It then searches its if-then rules to see whether data about flying might be stored there. Among other things it finds:

```
IF (INSTANCE-OF X BIRD) THEN
(CAN-PERFORM X FLYING)
```

This tells the inference engine that if the variable *x* is a thing called bird, it can fly. The inference engine substitutes Tweety for *x*, and ends up with:

```
IF (INSTANCE-OF
TWEETY BIRD) THEN
(CAN-PERFORM TWEETY
FLYING)
```

From this it draws the key conclusion:

```
(CAN-PERFORM
TWEETY FLYING)
```

This is, of course, the right answer, but that's not all there is to the job. The inference engine knows that most of its rules are merely "default true"—that is, they're usually reliable, but not always. At the same time Cyc is thinking about Tweety, it also looks through the rest of its knowledge base for any facts or rules that might help it confirm or disprove its conclusion. The scan turns up something suspicious: listed in the bird frame are two things labeled Penguin and Ostrich that can't fly. Always careful, Cyc gives the Tweety frame a second look to see if it is either a penguin or an ostrich. Finding no evidence that Tweety belongs to either of these categories, it lets its original conclusion stand: yes, Tweety can fly.

Tough luck, Sylvester.

With the help of this glowing, interactive work surface, Cyc programmers can use a stylus to move data and draw diagrams in the same way home computer operators use a mouse.

PHOTOGRAPH BY ERIC SANDER/GAMMA LIAISON

sponsorship, Lenat has managed to set up a working environment that might have been lifted from any graduate school AI department. Young people in T-shirts and sneakers lean against doorways with labels like HACKER'S HELL and catch each other up on late-night computing feats. At lunch, which is shared by the group every day, arguments about obscure points of theory are spirited and sometimes punctuated by flying napkins and lime wedges.

The air of informality aside, the task that confronts Lenat and his team of 12 is a rigorous one. Gathering knowledge isn't the problem. What's hard is finding a way to represent the knowledge so that a machine can make use of it. After all, Cyc can't learn about automobiles by going for a ride in the countryside. Each piece of knowledge must be carefully spoon-fed in exactly the right form. Lenat calls it teaching by brain surgery.

The surgery is carried out in a sort of programming language developed by Lenat and his collaborator R. V. Guha specifically for its ability to handle the fuzzy concepts and irregularities that characterize the real world. Called CycL (and spoken by CycLists), it is based in part on the concept of "frames." A frame is a collection of facts and rules that tells Cyc what it needs to know about a "unit," or a particular thing or concept. The frame describing the unit Doug Lenat, for example, tells Cyc that he is an example of the unit Person, that he is a professor, that he belongs to the Democratic party, and that among the people he likes is Cyc administrative director Mary Shepherd (in fact, he lives with her, but apparently no one wanted to get into this with Cyc).

On the computer screen a frame looks like a fact sheet; to build a frame about a particular unit, a Cyc "knowledge enterer" must first decide which kinds of information about the unit will be useful to the program. These information categories, or "slots," are then typed in as a list. The slots are then filled with "values." The frame for the unit Eating, for example, might look like this:

EATING
WHAT HAPPENS WHEN DELAYED: HUNGER
PERFORMED BY: ANIMALS
THINGS CONSUMED: FOOD
MORE GENERAL PROCESS: CONSUMING

The knowledge enterer must ensure that Cyc knows what all the slots and values mean. Thus Cyc needs to have a separate frame constructed for Things Consumed and Food (if those don't already exist) and for every other slot and value that appears in the frame.

The units that are described by the frames can often inherit traits from other units. Thus, because the unit Chewing is an example of the unit Bodily Function, Cyc automatically knows that chewing is one of the things that a person or animal is likely to do on a regular basis. Similarly, if Cyc knows that Bob and Jim are neighbors and that Bob's house is in Texas, it will know that Jim's house is probably there also.

The sentence "Napoleon died in 1821; Wellington was saddened" required a two-month explanation of life, death, and emotions.

Cyc's ability to process several frames at one time helps it draw other relationships between units. If one of Cyc's frames tells it that World War II took place between 1939 and 1945, and another tells it that a particular battle between two of the war's combatants occurred in 1943, it will correctly conclude that the smaller clash was part of the larger war.

A portion of Cyc's programming is set aside for processing logical constructions. If told that "Bill is either a terrific fisherman or a terrific liar," Cyc understands this to mean that the unit Bill has one or the other of these attributes, but not both. Cyc then consults what it knows about fishing, lying, and perhaps even egos and understands that fishermen like Bill may sometimes be motivated to fib about what they've caught.

Like most AI programs that have to deal with the real world, Cyc uses default logic, which means that it can be told to assume something is usually true but that there may be exceptions. Thus, rather than having to specify to Cyc that horses, mice, people, and elephants give birth to live young, a programmer can simply inform it that almost all mammals give birth to live young. If Cyc then runs across a reference to a platypus and learns this animal too is a mammal, it will guess that the platypus follows the birth rule; someone would have to tell Cyc that the platypus lays eggs. That may seem inexact, but, in fact, people reason this way, too.

"If I asked you where your car was, you would assume that it was where you left it," says Lenat. "Your car may have just been stolen, but it's better to get an occasional surprise than to exhibit the irrational behavior that would result from always thinking that your car was stolen."

Like we do, Cyc also has to know how to ignore irrelevant information, lest it waste too much time thinking or providing answers that are correct but useless. "When we're deciding whether to lock our car, we don't ask ourselves how many humps a camel has or what we had for dinner," says Lenat. "And if someone asked if there had recently been a large quantity of blood in your car, you would know not to take into account the fact that your body is a big bag of blood."

Cyc has ways of avoiding such impracticalities. It knows, for example, to start its search with the most obviously relevant frames, and then to branch out to those frames that have the most in common with the original frames. Thus, if asked a question about freeways, Cyc might start with the frame for freeway, then move on to roads, cars, and so on, coming across its answer long before it got to railroads and camels. In the case of the blood question, Cyc would take advantage of its knowledge that in many situations people overlook the fact that they contain blood, allowing the computer to put that fact near the end of its list.

To improve its speed further, Cyc is given access to about two dozen different types of reasoning, each of which is best suited for a certain type of problem. For complex problems, Cyc may sift through thousands of rules to reach an answer. But if asked how grandparents are related to their grandchildren, Cyc may simply apply inverse reasoning

to its knowledge that grandchildren are their grandparents' children's children and conclude that grandparents are their grandchildren's parents' parents.

When information is incomplete or uncertain, Cyc can try to fill in the blanks and assess plausibility to come up with a best guess. If told that a person has voted and that the person's mother is 40 years old, for example, Cyc might guess that the voter is about 19 or 20. You've got to be at least 18 to vote, Cyc knows, but if you're much more than 20 you're not likely to have a 40-year-old parent.

Bringing Cyc to this level of sophistication was an extraordinarily painstaking process. For the first year of the project Lenat and his team hardly went near a keyboard. Why bother trying to teach Cyc what a truck is if it doesn't understand cars, wheels, or even motion? The group had to start by defining such fundamental concepts as space, time, action, and physical objects.

"We sat around for four months with enormous pieces of white paper," recalls Lenat, "trying to determine what categories were worth distinguishing. A lot of our insights were surprising, like when we realized that what we think of as tangible objects, such as a person or a table, are also events with a starting time, an ending time, and a duration. We don't like to think of it, but people and things are executing the transitory process of existence."

The team also found it necessary to teach Cyc the complicated notion that people and objects have physical and nonphysical components that must be regarded separately. If asked to provide the age of a book, for example, Cyc would have to determine whether it should consider the date that the physical book itself rolled off the presses or the date that the contents of the book came into being—that is, when the author finished the manuscript.

Many other abstract representations were programmed into Cyc in those first years. Time was explained in terms of 50 different types of relationships between events, such as, "Event A starts after the end of event B." Physical objects were broken down into things that are countable, like marbles, and things that are "stuff," like peanut butter.

To expand this basic library of concepts, Lenat sometimes picked random sentences and tried to figure out everything the computer would need to know to understand them—much the same process Pittman and the other programmers use today. As it turned out, Cyc needed to know a lot. One of the first sentences—"Napoleon died in 1821; Wellington was saddened"—required Lenat's spending two months explaining such global concepts as life, death, communication, and human emotions before the seven words made any sense to the computer.

"It seemed like a never-ending process," says Lenat, but when the programmers were finally through, Cyc had been given a basic version of what AI researchers call an ontology, or a worldview, through which it could make sense of reality. It was only this past year that Lenat's group began to pour in the millions of bits of knowledge that make up that reality. Now each bit of data Cyc records makes processing the next bit that much easier. "If we want to teach Cyc what a cheetah is," Lenat says, "we just call up what it knows about lions and tigers and make a few changes."

The data base now contains 2 million pieces of knowledge; although that's a long way from 100 million, Lenat believes that with Cyc's increasing speed, four years is plenty of time to knock off the remaining 98 million.

Cyc's growth is partly due to Lenat's efforts to find people who are good at analyzing and entering knowledge. Most people, he found, simply don't have a knack for it. Eventually Lenat came across Pittman, who is a botanist by training, and Nick Siegel, a cultural anthropologist. "Doug wanted people with some cross-cultural experience, who could examine the implicit knowledge in society," explains Siegel. There are now three dozen people who enter information into Cyc on at least a part-time basis, but Pittman and Siegel are the only full-time knowledge enterers and are responsible for much of what Cyc knows.

Every day Pittman, Siegel, and others read of a wide variety of publications (including this one) and ask themselves what Cyc would need to know to understand them. A group favorite is the

Every day the Cyc team scans newspapers and magazines, looking for the commonsense knowledge that Cyc must have to understand the stories.

PHOTOGRAPH BY ERIC SANDER/GAMMA LIAISON

World Weekly News. "What are the things we know," muses Lenat, "that allow us to reject as untrue an article about the discovery of a human skeleton on the moon?" After teaching the new knowledge to Cyc, team members ask it questions to see if the lesson has sunk in. Having been told that someone drove to work in the morning, for example, Cyc might be asked how that person got home that evening. If Cyc answers "by boat," something obviously went wrong.

Cyc can also ask questions of its own to clarify points, resolve apparent contradictions, and fill in gaps. Displayed on the computer screen in plain English, the questions reflect a keen and curious mind. One recent inquiry: "Reproductions make sense for works of art, but someone is using the word to apply to something else besides a work of art. Is it okay for something that is partially tangible to have reproductions made of it?" One of Cyc's most poignant questions came up when it was told that intelligent things tend to like other intelligent things of the same type and then was later told that Mary Shepherd liked Cyc. "Am I a person?" asked Cyc. "Or is Mary Shepherd a computer program?"

At night Cyc is left alone to me[illegible]. That is, it looks over its knowledge base in search of interesting analogies. These analogies are examined in the morning by Lenat and others as a way of weeding out flaws. So far, Cyc has come up with a good number of imaginative associations. A recent night's ruminations included the following analogies: *dad* to *dictator* and *head of state; automobile* to *elevator, escalator,* and *consumer electronics; owns* to *physically contains* and *is aware of; profession* to *prominent for* and *occupant of.*

Although some of the analogies seem strained, the meanings are similar enough to indicate that Cyc absorbed its lessons. Often, however, the pupil misses the point. When Cyc analogized *United States* to *USA*, the programmers had to explain that the two terms were not analogous, but synonymous.

If Cyc can learn to digest plain English, the rate of growth of its knowledge base will explode. Instead of relying on a handful of people to type in selected crumbs of information, it will tear into everything from Descartes to *People*. But getting a computer to understand English or any other natural language is an elusive goal. Still, Lenat thinks he will succeed, and his reasoning is predictable: "You can't do natural-language understanding without at least a preliminary Cyc-type knowledge base."

Natural-language programs have generally focused on teaching computers grammar and syntax, under the assumption that if a machine knows something about the various types of words, meaning will follow. But, notes Lenat, pulling out the many possible meanings of a word in a typical sentence can be a herculean task. "What is a 'red conductor?' " he asks. "Is it a piece of metal that's red? A Communist musical conductor? Or take the two sentences, 'Mary saw a bicycle in the store window. She wanted it.' How do we know that she wanted the bicycle, and not the store or the window?" Things get even stickier in the real world, where Cyc would have to deal with newspaper headlines like BRITISH LEFT WAFFLES ON FALKLANDS.

If Cyc learns to digest plain English, it will be able to tear into everything from Descartes to *People*.

Cyc's natural-language interface is being designed by a group of programmer's down the hall from Lenat. The interface will work in conjunction with Cyc to analyze an English sentence for all its possible meanings and then select the one that best jibes with its common sense. There are some early signs that the intended synergy is paying off: when recently informed that someone had "read Melville," Cyc guessed the person had read *Moby Dick*, even though it had never been told the meaning of the expression "to read an author."

But Cyc could fail before it ever gets a chance to read. The key danger, says Lenat, comes from divergence—that is, from different pieces of knowledge conflicting with or obscuring one another. A certain amount of divergence is inevitable. Different people are bound to provide Cyc with different ways of looking at the same thing, such as when Cyc is told by one person that *father* is a man who begets a child and by another that it is a male parent. Lenat insists that most inconsistencies can be caught through Cyc's nighttime analogy hunts. Even if some slip through, he points out, Cyc's ability to recognize the fallibility of the "truths" in its knowledge base prevents it from crashing.

However, Lenat concedes that the growth of the knowledge base could conceivably outstrip this protection mechanism and allow fatal divergence to set in. "If there are too many inconsistencies," he says, "the knowledge base will collapse." With all its energy devoted to reconciling contradictions, Cyc would lose the ability to do anything else. For now, however, Cyc's knowledge enterers have been able to stay in agreement—and this keeps Lenat hopeful.

Others are less confident, claiming that Lenat has done a better job raising expectations for Cyc than he is likely to do building it. There are many thorny AI problems that have to be solved, critics say, before he can achieve the kind of results he's after. For example, Cyc needs a better way to distinguish between statements of fact like "Jupiter is the largest planet in the solar system" and statements of opinion like "The Reds are the best team in the National League."

Even Lenat's friend and supporter Marvin Minsky, famed MIT professor and artificial-intelligence pioneer, expresses some worry about Cyc's approach to knowledge. "Cyc has a rather logical structure," he notes. "Lenat is trying to make it more flexible with frames, but it's still a single way of representing knowledge, and no one representation will work well. I think the systems of the future will have two or three different ways of representing knowledge with cross-links between them. That's how the brain works: one part has knowledge about people, another about how things work, and so on, to hundreds of specialized areas."

Most researchers seem to think that Lenat's most unrealistic prediction is that Cyc will read English within five years. Minsky too is skeptical, but he

insists it isn't an issue worth debating. "Who cares if it takes twenty years," he asks, "as long as it works?"

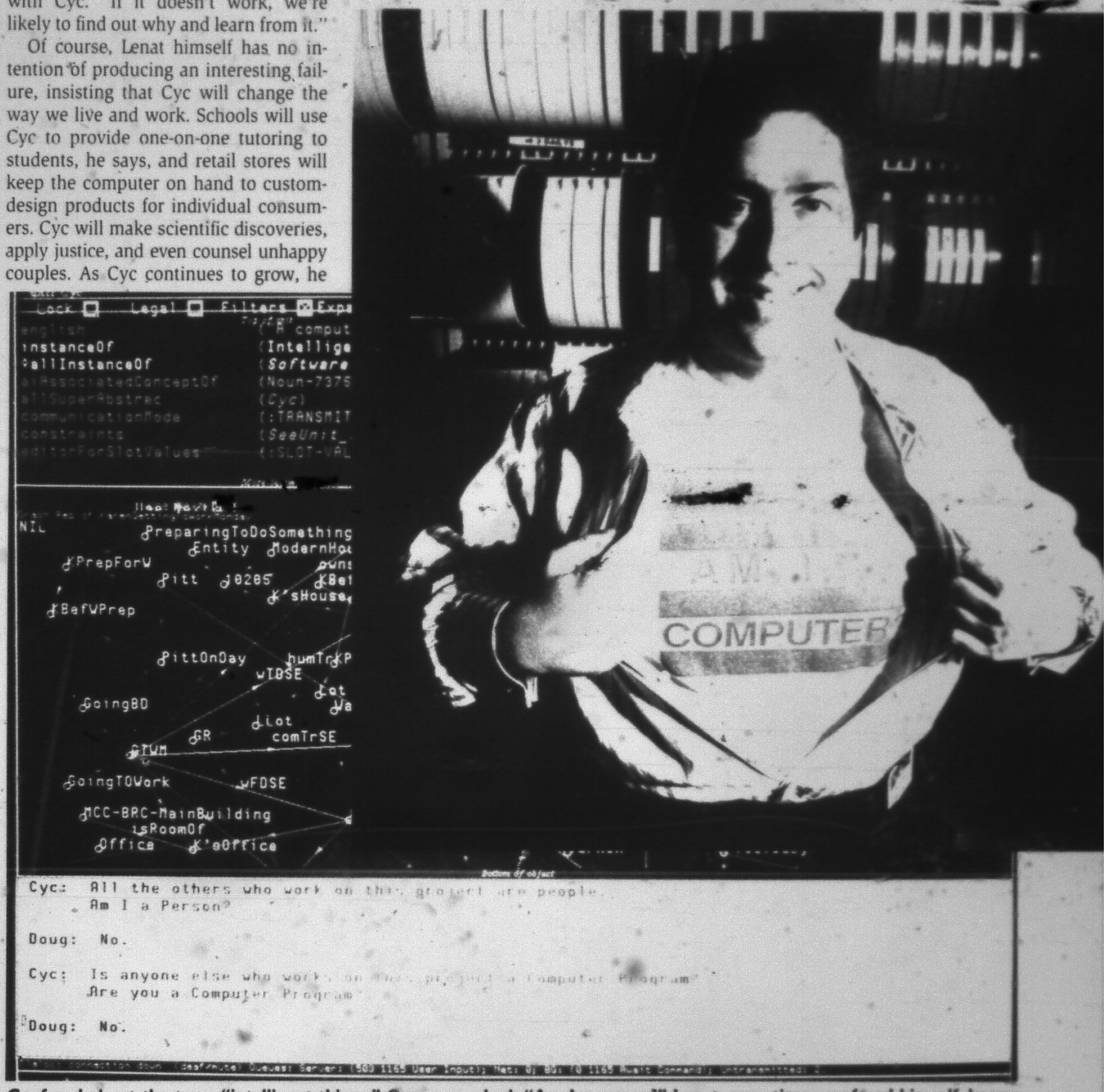
In fact, AI researchers are quick to point out that even in failure Cyc would be a huge boon to the field. "This is one of the riskiest projects in AI, but it's also one of the most interesting and visionary," says Fanya Montalvo, a researcher at Digital Equipment who is working with Cyc. "If it doesn't work, we're likely to find out why and learn from it."

Of course, Lenat himself has no intention of producing an interesting failure, insisting that Cyc will change the way we live and work. Schools will use Cyc to provide one-on-one tutoring to students, he says, and retail stores will keep the computer on hand to custom-design products for individual consumers. Cyc will make scientific discoveries, apply justice, and even counsel unhappy couples. As Cyc continues to grow, he predicts, pieces of it will be stored in computers around the world, its contents made available through phone lines and radio waves. Cyc's intelligence, he claims, will flow like electricity through a gigantic, ubiquitous knowledge grid.

Clearly, Lenat is not a man plagued by self-doubt. As for others' doubts, he chalks some of it up to our misplaced need to attribute something unique to human intelligence. "It's normally nice to mystify the way people reason," he says. "But it's even nicer for us to demystify it. We're not as inscrutable as we think." ❑

Contributing editor David H. Freedman wrote July's cover story on physicist Sidney Coleman and parallel universes.

PHOTOGRAPHS TOP BY TERRY VINE, BOTTOM BY ERIC BANDER-GAMMA LIAISON



Confused about the term "intelligent things," Cyc once asked, "Am I a person?" Lenat, sometimes confused himself, knows that the answer is not always easy. The data road map in the background is a "semantic net," used by programmers to enter information into Cyc. This net tells Cyc how programmer Karen Pittman (represented as "Pitt") drove to work.